A Survey on File Defragmentation Techniques on Modern Storage Systems

Sangjin Lee*, Lan Anh Nguyen*, Hyeongi Yeo*, Sunggon Kim†, Sungrae Cho*, and Yongseok Son*
*Department of Computer Science and Engineering, Chung-Ang University
†Department of Computer Science and Engineering, Seoul National University of Science and Technology

***Abstract*—File fragmentation is one of the widely studied subjects for several decades due to its considerable degradation on I/O operations. Also, it is a well-known problem in academia that file fragmentation on HDD disks can extremely lower the throughput and increase tail latency. However, even with modern storage devices, such as flash-based SSDs and Optane SSDs, the highly fragmented file systems shows considerable underutilization of the storage device. Due to the different nature of the modern storage systems in comparison to the conventional disks, the fragmentation is often called 'aging'. This paper shows an overview of the aging of modern storage devices and their fragmentation effects. Also shows the survey of file fragmentation and its related literature, showing the studies discovering fragmentation in file systems, and mitigation techniques which suitable for modern storage devices.**

***Index Terms*—Fragmentation, File system aging, Defragmentation**

I. Introduction

SSD device is commonly chosen as secondary memory for storage sub-systems in I/O intensive systems due to their high performance, resulting in taking the place of rotating disks(HDDs). Especially, modern storage devices with multiple internal buses and internal physical buffers (e.g., DRAM) have been widely adopted for enterprise cloud storage or cluster servers to support evergrowing data processing load induced by data-intensive applications. The increasing throughput and low latency by leveraging the internal parallelism of modern storage devices have distinct advantages in comparison to the conventional HDD disk, and the fragmentation of the file system is known to be a solved or negligible problem.

However, SSD-backed storage sub-systems could suffer from file fragmentation as the file operations(create, delete, move, truncate). Even systems like a data center or a cloud service that have to deal with a large amount of I/O operations coming in between multiple applications can suffer from high fragmentation due to the increased overhead driven by scattered blocks all across non-contiguous block addresses. For example, the EXT4 file system has its designated defragmentation tool called `e4defrag` and it could improve the overall performance of the file system by re-allocating the dispersed blocks into contiguous block space. But the process is done with a single thread, which could be adequate for a conventional HDD device, hindering the overall defragmentation process and making maintenance of the storage slow. Also, the single I/O request from the application tends to amplify into multiple small I/Os.

In this paper, we investigate studies on file fragmentation, especially for modern storage devices(i.e., flash-based SSD, Optane SSD). We present an existing defragmentation process of the EXT4 file system to enhance further knowledge on file defragmentation. Furthermore, we conduct a comparison in terms of the reason for fragmentation and mitigation methods for fragmentation. The rest of this paper is organized as follows. Section II describes fragmentation on file systems and SSDs. Section III presents the literature which explains the cause of the fragmentation, and studies with proposals on efficient defragmentation for modern storage systems. Finally, Section IV concludes this article.

II. File Fragmentation

If the file system has become highly fragmented, the logical block device cannot handle I/O requests from the application layer written on contiguous block space. The file system needs to allocate a single file to the number of divided extents more frequently than before, resulting in performance decrease.

A. Fragmentation on SSDs; aging

Fragmentation on traditional HDD disks leads to a persistent drain on overall I/O performance [1] due to the inherent mechanical seek time involved. To address the issue, regular defragmentation processes (i.e., `e4defrag`) are essential for avoiding the suboptimal performance of HDD-equipped subsystems.

In contrast, flash-based storage devices do not require further mechanical seek operation. Therefore, it is generally believed that file systems utilizing SSD devices do not necessitate additional defragmentation processes. But even on flash-based SSDs, deterioration in logical block locality (non-contiguous logical blocks) can still have a negative impact on performance [2], [3]. Several papers show that non-HDD storage devices suffer from various reasons. The amplification of the I/O request, under-optimized kernel storage I/O subsystem, recurring problem on mobile flash storage, and even free-space(the unused disk blocks) aging can be the reason for under-utilization while aging of modern SSD devices [4]–[6]. Furthermore, as tools used to invoke aging experimentally [7] are continuously being studied, file system fragmentation(aging) in SSDs is an important problem.

Further knowledge and details could be found on Section III.

B. Defragmentation in EXT4 file system

Defragmentation reduces file fragmentation by gathering dispersed blocks of a file and repositioning the file accordingly. Especially, the EXT4 file system, which is in the mainline of the Linux kernel, uses the defragmentation process specifically designed for itself(e.g., `e4defrag`). The `e4defrag` is performed within both user space and kernel space. In the user space, a single defragmentation worker collects the target file-related information (e.g., inode) in the user space, and determines the fragmentation state of the target file. If the file is not fragmented, the defragmentation thread simply skips the target file. But, if the file needs defragmentation the worker creates a temporal file the same sized as the target file (fragmented file) with contiguous blocks. Next, the worker moves the blocks in the target file to the new file. During the relocation of the target blocks of the new file, the `EXT4_IOC_MOVE_EXT()` system call is employed. The defragmentation worker read all the dispersed blocks to the page cache and exchanges the block mapping information of the files. Finally, unlink the target file and the new file. All of the defragmentation processes in `e4defrag` are performed by a single defragmentation worker, which was suitable for traditional spinning disks.

III. Defragmentation in literatures

In current literature, file system fragmentation(i.e.,aging) and methods for mitigating the fragmentation (i.e., defragmentation, anti-aging) are widely studied. In this section, we will discuss studies related to fragmentation.

Hahn et al. [5] investigate and evaluate file fragmentation on mobile flash devices. They proposed decoupled defragmenter, janusd, which could support logical and physical defragmentation. They observed that on mobile flash devices, the file fragmentation can cause recurring problem, and also the file fragmentation on mobile flash storage has a different influence in comparison to traditional HDD disk.

The fragmented state of the file system, as referred to by Conway et al. [2], is termed age. They analyze prior works on file system aging as artificial aging technique, aging measure, and age mitigation. The paper's evaluation has been done on various file systems, including EXT4, ZFS, XFS, F2FS, Btrfs, and BetrFS. These evaluations are conducted by using both micro and realistic workloads, specifically focusing on the "git pull" operation, and were carried out on both HDD and SSD. As a result, they demonstrate that BetrFS outperforms other file systems in terms of aging performance within the workloads, and inspect key features of BetrFS aging avoidance.

Conway et al. [6] investigate the prevalent assumption that, increased space pressure contributes to the exacerbation of the file system fragmentation. However, the study reveals a mere 20% reduction in subsequent read speeds on EXT4. By this result, they indicate that the impact of free-space fragmentation on read performance can be characterized as expediting the aging process of the file system. As a discussion, the paper seeks to stimulate discussion by challenging the commonly held notion that disk fullness has a direct and significant impact performance of file systems.

The paper [8] targets log-structured file system, especially F2FS, and effectively eliminates file fragmentation by up to 98.5%. But the proposed scheme cannot remove fragmentation when valid blocks are scattered across the file offset. They make up for weaknesses and propose an anti-aging log-structured file system called AALFS [9] by leveraging observations perceived by their previous research. The paper extensively analyzes the performance degradation incurred by file system fragmentation and kernel-level overheads with various storage types(i.e., HDD, Micro SD(flash storage), flash-based SSD, and Optane SSD). The proposal re-arranges the order of valid blocks to eliminate file fragmentation by using file offset and inode number. As a result, AALFS achieves x22.8 of I/O performance compared to the fragmented file system equipped with an HDD disk. Experiments focused on IOzone sequential read with various request sizes, and on real-world database, SQLite.

Fragpicker [4] devises defragmentation scheme for modern storage devices, such as Flash-based SSD, and Optane SSD. They target the reduction of the amount of multiple I/Os derived by a single I/O request, which is called request splitting. The authors expose the issue that has been driven by the current Linux kernel I/O call stack, which is not capable of showing non-contiguous areas in terms of LBA (Logical Block Address). Consequently, their observations indicate that modern storage devices experience a deterioration of the performance while management of I/O requests, is driven by current Linux kernel I/O stack overhead. This leads to an escalation of the interface overhead between the host operating system and the storage device, resulting in impeding the utilization of device resources.

As a follow-up study, the paper [10] additionally evaluate FragPicker with YCSB workload-C. They set up RocksDB on EXT4 equipped with Optane SSD. Their experiment shows that the execution time of the defragmentation decreased to 16% of the `e4defrag` and reduce the performance degradation of co-running processes by 45%. They show a thorough examination of the performance and fairness degradation resulting from fragmentation, taking into account storage device internals.

Zhu et al. [11] propose epdefrag to accelerate the overall defragmentation process by exploiting SSD internal parallelism by designing multiple threads to perform the defragmentation process for multiple files in parallel. The epdefrag is based on `e4defrag` but relocates files in a parallel manner, and their experimental result shows x2.96 reduced execution time compared with an existing scheme.

From the perspective of I/O control, the paper [12] by Park and Eom studies the performance of a fragmented F2FS file system among simultaneously running multiple applications. Their experiment shows the fragmentation state causes an increasing number of I/Os(splitting an I/O into smaller-sized I/Os) hindering, to the conventional I/O control mechanisms such as CFQ(Completly Fair Queuing), BFQ(Budget Fair

Queuing), and even with new I/O control model IOCost [13]. The authors have evaluated the negative influence on F2FS while fragmented state and investigated homogeneous and heterogeneous workloads with the defragmentation tool Fragpicker. Finally, the paper has found that with the fragmentation of F2FS, various I/O control mechanisms fail to achieve their scheduling goals for different reasons, and can be relieved by defragmentation on F2FS.

Kesavan et al. [14] reported that fragmentation of the free-space in the file system can lower the write performance as well as read operation. They believe that the file systems with sub-block granular addressing can gather intra-block fragmentation and can successfully prevent file system fragmentation. By the hypothesis, the paper proposes a NetApp® WAFL® file system that leverages the storage virtualization instances, which can relocate PBA (Physical Block Address) efficiently and quickly. However, due to the configurational difference between file systems and WAFL, they provide inherent trade-offs by historical context. As a result, they present storage gardening techniques leveraging the FlexVol® virtualization layer to remedy the fragmentation.

IV. Conclusion

Regarding the importance of fragmentation state on file systems in the field of storage I/O performance, and to exploit the computing power of ever-growing high-performance SSDs, the file system fragmentation state and its mitigation are actively researched by a number of literature.

In this survey, our main focus has been centered on describing each study, conducting comparisons, and analyzing the distinct attributes within each literature. Through this paper, we have been able to propose guidelines aimed at researchers interested in file fragmentation and building background knowledge in related topics.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT), KIAT grant funded by the Korea Government (MOTIE), and Korea Institute of Science and Technology Information (KISTI). (No. NRF-2021R1C1C1010861, NRF-2022R1A4A5034130, KIAT-P0012724, RS-2022-00166541) (Corresponding Author: Yongseok Son).

References

[1] K. A. Smith and M. I. Seltzer, "File system aging—increasing the relevance of file system benchmarks," in *Proceedings of the 1997 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pp. 203–213, 1997.

[2] A. Conway, A. Bakshi, Y. Jiao, W. Jannen, Y. Zhan, J. Yuan, M. A. Bender, R. Johnson, B. C. Kuszmaul, D. E. Porter, *et al.*, "File systems fated for senescence? nonsense, says science!," in *15th USENIX Conference on File and Storage Technologies (FAST 17)*, pp. 45–58, 2017.

[3] C. Ji, L.-P. Chang, L. Shi, C. Wu, Q. Li, and C. J. Xue, "An empirical study of {File-System} fragmentation in mobile storage systems," in *8th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 16)*, 2016.

[4] J. Park and Y. I. Eom, "Fragpicker: A new defragmentation tool for modern storage devices," in *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, pp. 280–294, 2021.

[5] S. S. Hahn, S. Lee, C. Ji, L.-P. Chang, I. Yee, L. Shi, C. J. Xue, and J. Kim, "Improving file system performance of mobile storage systems using a decoupled defragmenter," in *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, pp. 759–771, 2017.

[6] A. Conway, E. Knorr, Y. Jiao, M. A. Bender, W. Jannen, R. Johnson, D. Porter, and M. Farach-Colton, "Filesystem aging:{It's} more usage than fullness," in *11th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 19)*, 2019.

[7] S. Kadekodi, V. Nagarajan, and G. R. Ganger, "Geriatrix: Aging what you see and what you {don't} see. a file system aging approach for modern storage systems," in *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pp. 691–704, 2018.

[8] J. Park, D. H. Kang, and Y. I. Eom, "File defragmentation scheme for a log-structured file system," in *Proceedings of the 7th ACM SIGOPS Asia-Pacific Workshop on Systems*, pp. 1–7, 2016.

[9] J. Park and Y. I. Eom, "Anti-aging lfs: Self-defragmentation with fragmentation-aware cleaning," *IEEE Access*, vol. 8, pp. 151474–151486, 2020.

[10] J. Park and Y. I. Eom, "Filesystem fragmentation on modern storage systems," *ACM Transactions on Computer Systems*, 2023.

[11] G. Zhu, J. Lee, and Y. Son, "An efficient and parallel file defragmentation scheme for flash-based ssds," in *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, pp. 1208–1211, 2022.

[12] J. Park and Y. I. Eom, "File fragmentation from the perspective of i/o control," in *Proceedings of the 14th ACM Workshop on Hot Topics in Storage and File Systems*, pp. 126–132, 2022.

[13] T. Heo, D. Schatzberg, A. Newell, S. Liu, S. Dhakshinamurthy, I. Narayanan, J. Bacik, C. Mason, C. Tang, and D. Skarlatos, "Iocost: block io control for containers in datacenters," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 595–608, 2022.

[14] R. Kesavan, M. Curtis-Maury, V. Devadas, and K. Mishra, "Storage gardening: Using a virtualization layer for efficient defragmentation in the {WAFL} file system," in *17th USENIX Conference on File and Storage Technologies (FAST 19)*, pp. 65–78, 2019.